VA Kidney App Architecture

Overview

Develop a mobile application that patients and caregivers can use to make smart kidney related nutrition decisions informed by their own laboratory/biometric data and personal goals.

This application will be for both iOS and Android operating systems.

General Workflow

User Profile

All users must agree to a Terms of Service/Liability Release prior to launching the app for the first time. The App shall permit users to personalize profiles with an avatar. User will input profile data and each profile should include the following:

- Name
- Age
- Height
- Current weight: Daily for all and pre/post dialysis and daily values for those in ESRD and on dialysis. On Dialysis Days, there must be a way to record 2 weights (pre/post dialysis). They won't enter their weight every day, perhaps, but they should be ABLE to enter it daily or >1 times/day.
- Disease category (early stage CKD, ESRD and change in kidney function over time)
- Presence or absence of diabetes and diabetic diet goals
- Lab values: eGFR, blood sugar, albumin, potassium, phosphorus, hemoglobin A1C,
- PTH, calcium, bicarbonate (CO2), uric acid, vitamin D
    - The units of measurement for each will be automatically listed next to the item.
- Blood pressure
- Medications
- Goals: Users will be able to create and track goals. For example: Lose weight/input desired weight, eat more add at least 2 cups of fresh fruits and vegetables to their existing diets daily if their remaining kidney function is sufficient to avoid hyperkalemia from the added potassium load , improve the lab levels, drink xx ounces of water/day

User History/Trend Metrics

App will have data calculators/trackers monitor and display past and current values for the following data input by the user:

- Labs: eGFR, Albumin, phosphorus, potassium, bicarbonate (CO2), uric acid, A1c.
- Dietary and fluid inputs: grams of protein, milligrams of potassium, sodium, phosphorus,
- milliliters or ounces of fluids
- Body weight pounds or kilograms
- Blood pressure

User Incentives (Gamification)

The App shall employ a fun and positive gamification incentive approach to promote personal goal-setting and achievements. The gamification, goal-setting and reward structure will be tailored to the user's disease category (early stage CKD, CKD, ESRD) allowable limits/requirements.

The features shall include (but not be limited to):

- Personal goal setting and prioritization
- Reward structure for specific achievements (for example: goal progress in lowering
- blood pressure or reducing body weight)
- The following data will be part of the gamification feature :
    - Labs: eGFR, albumin, phosphorus, potassium, bicarbonate (CO2), uric acid, A1c
    - Dietary and fluid inputs: grams of protein, milligrams of potassium, sodium, phosphorus

Intake and Activity Tracking

App shall permit dietary and fluid intake tracking.

- For daily dietary intake the user will list the foods and fluids they have consumed throughout the day. Patients with kidney disease track fluid intake (fluid is anything that is liquid at room temperature. For example, ice cream, Jello, pudding, milk, juice, etc.)
- Search interface shall include an auto-populate feature that dynamically incorporates matches as the user is typing

App shall include a feature that permits users to include photos of food

- The user can take photos of their meals and the app will store the photos according to time/date and type of meal (breakfast, lunch, dinner) to allow for dietary recall
- This will serve as a daily consumption food journal feature
- Storage of these photos should be in the OS's native app with the ability to utilize the photo app's cloud based storage as well.

External API Usage: App shall include an activity tracking component that includes user inputs as well as syncing of biometric data from devices such as a Fitbit and the Apple Watch.

Data Export

Initially, it is expected that the application will allow a user to generate and share a PDF report of their activity with their healthcare provider. This report will be stored locally and the user will be given the option to share the report via mail, message, etc.

In the future, it would be desirable for the user to have the ability to export/import their data as needed.

Nutrition Recommendations and Drug Interactions

External API Usage: The app shall recommend food items based on individual lab inputs and disease stage. It will alert the user to unsafe levels of nutrient content, provide food and drug interaction warnings, suggest food substitutions, and input this information into individualized decision matrices based on individual labs, goals, and preferences. The app will also produce a monthly report card provided as an educational tool for updating patients on current labs and recommended foods.

The app shall notify users of nutrient intake precautions they should take with certain drug interactions.

Technologies Used

User Interface

The developer must follow Human Interface Guidelines for iOS

Wireframes

The screen structure and navigation must follow the separately provided wireframes and PDF with related notes.

Accessibility

iOS app must support accessibility provided by iOS as much as possible as the solution to Section 508 Compliance. This must be done after applying design because all UI components should have accessibility attributes and complete support of accessibility can be done only after all UI components are finalized (images and videos are added, static texts are finalized). Hence, for demo app this can be skipped.

- Accessibility on iOS: https://developer.apple.com/accessibility/ios/
- Section 508 Compliance: https://mobile.va.gov/content/my-app-section-508-compliant

Architecture Notes

- The app will have no centralized server interface. The app will consume external APIs as noted.
- The app is for iPhones only and supports iOS 10+.
- Navigation flow and bar must follow iOS Guidelines. For example, *details* screen and forms should have back button instead of “Menu” button on left top corner. The demo app follows wireframe and in some cases it’s not possible to move back while having “Menu” button.

Model objects

Some of the model objects storing data for the app are defined in demo app in VAKidneyNutrition/Model group in Xcode project. There are a few groups of the model object:

- objects created by user and presented in UI as solid elements, e.g. Goal, Medicine, etc.;
- objects generated by events, e.g. Reward, Suggestion, etc.;
- objects representing measurements from external devices and created during synchronization;
- helpful objects supporting other main objects, e.g. MedicationTime, etc.

All objects representing data are stored in Core Data storage (see below).
First group of objects is clearly defined by the UI and is straightforward. Reward, Suggestions and other

generated objects appear in the app with the following ways:

- After user enters new data. For example, when new Food Intake object was added, then the related Goals are updated. The updated goals in turn can issue Rewards, and so on. This should be implemented as “manual trigger”.
- Other objects appear at a given time or periodically. For example, some Rewards may be issued at the end of the day/week/month depending on the other data. This should be implemented using scheduler. Unlike the previous one, this triggers can be launched while the app is in background. The app must implement Local Notifications as a solution to inform the user about related events. For example, if at the end of the week the app is not running it should generate Local Notification about new Reward (if issue). By opening the notification user must be moved to corresponding screen. Reminders should work the same way.
- Measurement data from different external sources may also be handled in background. The app must implement correct updates of Rewards, Goals and other dependent objects. Measurements data can be provided in real time if the app is in foreground. But, in most cases the measurements will be obtained periodically from HealthKit storage.

Project structure

- Xcode project should group view controllers, XIBs, model objects and other supporting files properly.
- Each group of screen must have a separated XIB file to reduce the loading time.
- View controllers should control UI only and all storage operations and logic must be delegated to “API” classes and utilities.
- View controllers are grouped according to the XIB files inside *View Controllers* group.
- Shared helpful methods must be defined either in *HelpfulFunctions.swift* or *UIExtensions.swift*.
- Sample and static lookup files must be stored in *Supporting Files/Sample Data* group as JSON files.
- Dynamic UI components used in XIBs must be implemented in a separated framework. *UIComponents*. This allows to see the rendering in XIB without launching the app.
- Persistence service classes must follow already added services in *API/Caching* group. They use a pair of objects (model object, e.g. Goal and Core Data model object, e.g. GoalMO) and provide helpful methods if needed.

Scheduler and Notifications

The utility that issues generated objects (Rewards, Suggestions) must be implemented properly:

- Should have simple function that triggers all verifications and generates all required objects. This is required to easy debug and verification.
- Update data in foreground and in background. Timer class can be used while the app is in foreground. For background check, the app must implement background tasks.
- Local Notifications must be used to inform user about any updates. This is required because in

large amount of data it's hard to find the updates. The notifications must appear inside the app if the app is in foreground and in Notification Center if the app is in background. User must be able to turn on/off different types of notifications.

- The app must be protected from issuing extra rewards if minor changes are added. For example, user can modify a goal slightly after the reward issued and after adding more data the goal is "reached" again. All such cases must be properly handled.

Data Sources

Core Data as storage

Core Data must be used to store all entered data by the user except image/video data (see below).

Storage for media files

Images and videos can be temporary cached, but persistently stored in external storage to the app, e.g. CloudKit, Photos framework or Amazon's S3 bucket. This is needed to spend less space for the app and in the case of CloudKit and Photos framework to allow user to control the memory usage via Files app or Photos app. If user will have a control to remove the added photos to free the space on the device, then the related model objects in the app will be shown without images after crean up. However, we can store two versions of the images - one with very low quality that can be stored in Core Data and cannot be removed without removing the model object or uninstalling the app, and the second with high quality (in PhotoLibrary, iCloud or Amazon's S3). If user will clean up high quality photos we will show low quality thumbnails. This need to be decided by the client what storage to use. For this demo we store photos in Core Data.

- CloudKit: https://developer.apple.com/icloud/
- Photos framework: https://developer.apple.com/documentation/photos
- Files app: https://support.apple.com/en-us/HT206481
- Amazon S3: https://aws.amazon.com/ru/s3/

HealthKit and ResearchKit

The app must use Health Kit to store data provided by the user and other related information as much as possible. This framework was introduced by Apple exactly for such kind of apps. For demo app we can use plain objects, but in future challenges they must be equipped/replaced with model objects from Health Kit.

Also the app can support ResearchKit if it's reasonable and possible

- HealthKit: https://developer.apple.com/healthkit/
- ResearchKit: https://developer.apple.com/researchkit/

USDA Nutrient Database

Main Site: https://ndb.nal.usda.gov/ndb/

API Doc: https://ndb.nal.usda.gov/ndb/doc/index

NOTE: It lacks a way to customize nutrient content of a unique ethnic food, so we need to add this feature. (Example – an entry for Kandai Paneer could be made to combine nutrients from the paneer cheese, tomato and save for reentry).

FDA Drug Interaction and Product Labeling Database

Main Site: https://open.fda.gov/

API Doc: https://open.fda.gov/drug/